IP Telephony: Packet-based Multimedia Communications Systems

By Olivier Hersent, David Gurle, Jean-Pierre Petit

....
Publisher: **Addison Wesley Professional**
Pub Date: **December 09, 1999**
Print ISBN-10: **0-201-61910-5**
Print ISBN-13: **978-0-201-61910-2**
Pages: **480**
Slots: **2.0**

Copyright

PEARSON EDUCATION LIMITED

Head Office:

Edinburgh Gate

Harlow CM20 2JE

Tel: +44 (0)1279 623623

Fax: +44 (0)1279 431059

London Office:

128 Long Acre

London WC2E 9AN

Tel: +44 (0)207 447 2000

Fax: +44 (0)207 240 5771

Website: www.awl.com/cseng

First published in Great Britain 2000

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library.

Library of Congress Cataloging in Publication Data

Applied for.

1 0 9 8 7 6 5 4 3 2 1

Typeset by Pantek Arts, Maidstone, Kent.

Printed and bound in the United States of America.

The publishers' policy is to use paper manufactured from sustainable forests.

The origin and purpose of SIP

SIP is defined in RFC 2543 (March 99) of the MMUSIC (Multiparty Multimedia Session Control) working group of the IETF.

The MMUSIC working group is focused on loosely coupled conferences as they exist today on the MBONE (*see* Chapter 8 *for additional details on the MBONE*) and is working on a complete framework based on the following protocols:

- The Session Description Protocol (SDP) and the Session Announcement Protocol (SAP);
- The Real-Time Stream protocol (RTSP) to control real-time data servers;
- The Simple Conference Control Protocol (SCCP) for tightly coupled conferences. This work has just begun with the definition of a message bus between conferencing systems;
- SIP.

So far only SIP, RTSP and SDP have become requests for comments (RFCs).

These protocols complement existing IETF protocols, such as RTP from the AVT (Audio/Video transport) working group for the transfer of isochronous data, or RSVP from the INTSERV (integrated services) working group for bandwidth allocation.

Overview of a simple SIP call

Successful call to an IP address directly

Here we assume that the initiator of the call knows the IP address of the called endpoint (we will see later that there are many other types of SIP addresses). The caller might be calling the following SIP address:

sip: *john@192.190.132.31*

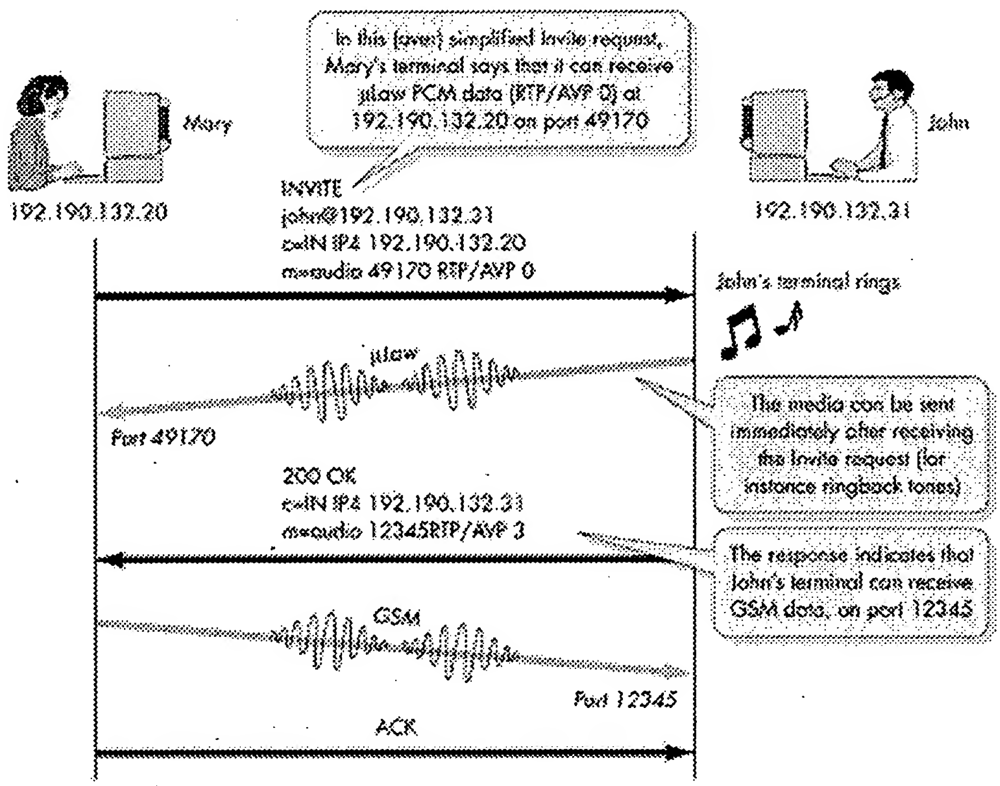
SIP entities communicate using 'transactions'. SIP calls a transaction a request (e.g. the Invite request below) and the

response(s) it triggers (200 OK in our example) up to a final response (see the definition below, all 2xx, 3xx, 4xx, 5xx and 6xx responses are final). The initiator of a SIP request is called a SIP client, and the responding entity is called a SIP server. The messages exchanged during a transaction share a common Cseq number, with one exception: the ACK message uses the same Cseq as the transaction to which it applies but is a transaction of its own.

The first step is to open a signaling connection between the calling and the called endpoint. SIP endpoints can use UDP or TCP signaling – the message syntax is independent of the transport protocol being used. When using TCP, the same connection can be used for all SIP requests and responses (*not* for the media data), or a new TCP connection can be used for each transaction. If UDP is used, the address and port to use for the answers to SIP requests are contained in the 'via' header parameter of the SIP request. Replies must not be sent to the IP address of the client. If no port is specified in the SIP address, the connection is made to port 5060 for both TCP and UDP.

A SIP client calls another SIP endpoint by sending an Invite request message (Fig. 2.1). The Invite message normally contains enough information to allow the called terminal to immediately establish the requested media connection to the calling endpoint. This information includes the media capabilities that the calling endpoint can receive (and send: coder capabilities are assumed to be both sending or receiving in SIP, unless the SDP parameters 'sendonly' or 'reconly' are used), and the transport address where the calling endpoint expects the called endpoint to send this media data. Most endpoints will be able to receive many different encodings for each media type. The particular encoding chosen by the sender appears as part of the RTP header.

Figure 2.1. Making a call to a known IP address



The called endpoint needs to indicate that it is accepting the request. This is the purpose of the OK response message. Since the request was an invitation, the OK response also contains the media capabilities of the called endpoint, and where it is expecting to receive the media data. The caller needs to acknowledge that it has properly received the response of the called endpoint (remember we might be using UDP) with the ACK message.

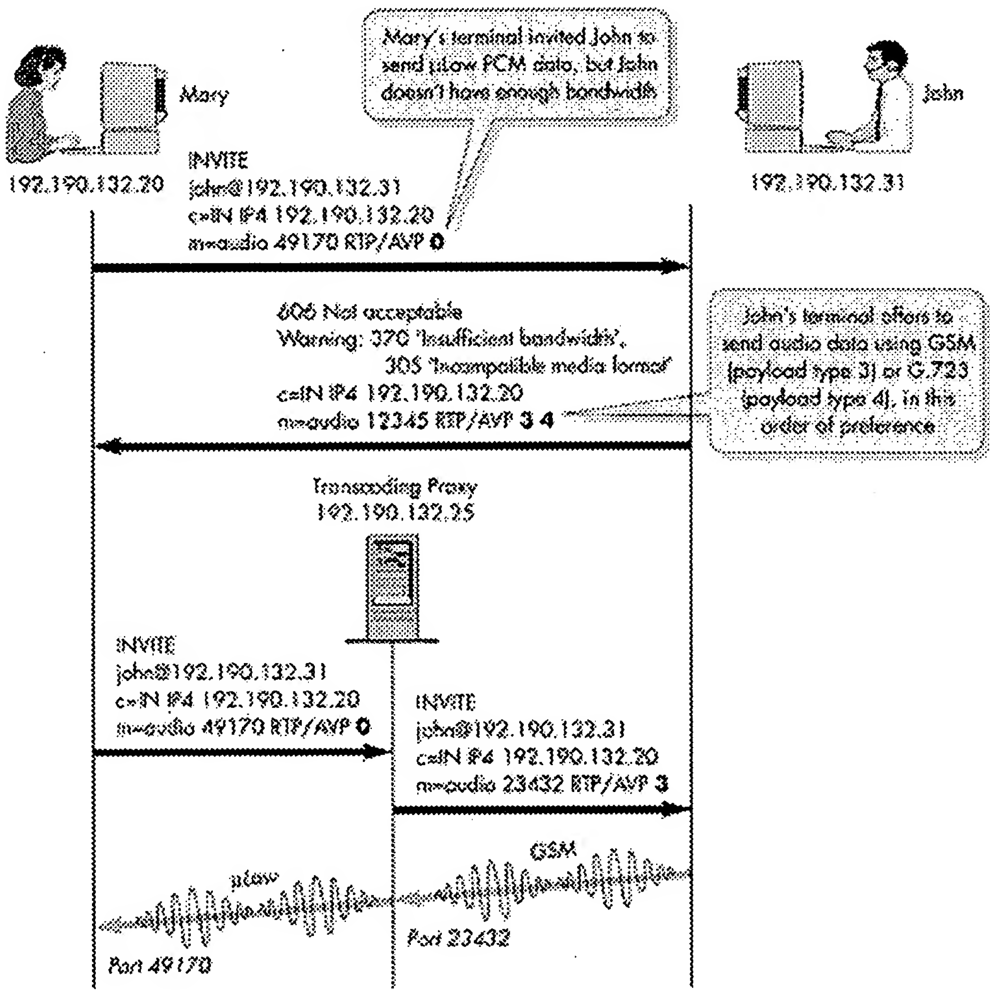
From this simple example, we can see that SIP is very efficient – the callee to caller media channel can be set in one round trip, and the caller to callee media channel can be set up in one-and-a-half round trips. This is much better than the

many round trips needed by the bootstrap nature of H.323v1 (H.323v2 is as efficient as SIP if fastStart call setup is used).

Codec negotiation

In the previous call, Mary's terminal offered to receive an audio channel encoded in PCM. This may not be acceptable for John's terminal, either because John does not have enough bandwidth available (PCM requires 64 kbit/s, plus the RTP/UDP/IP/PPP overhead) or because John's terminal does not have a PCM µ-Law coder. In this case John's terminal will reply with a 606 Not Acceptable reply, and eventually list the set of coders that it can use. With this information, Mary's terminal can either send a new Invite request, with the same call identifier, advertising the proper code (but if it had such capability, it could have sent it as a choice in the first Invite request) or re-initiate a call through a transcoding proxy. *See* Fig. 2.2.

Figure 2.2. Negotiating the call setup



The voice over IP forum agreed on a default coder to use for low bitrate voice, G.723.1, in order to keep the probability of such incompatibility to a minimum in H323. No such recommendation exists for SIP, but most terminals seem to be able to receive PCM A-Law and µ-Law, and GSM.

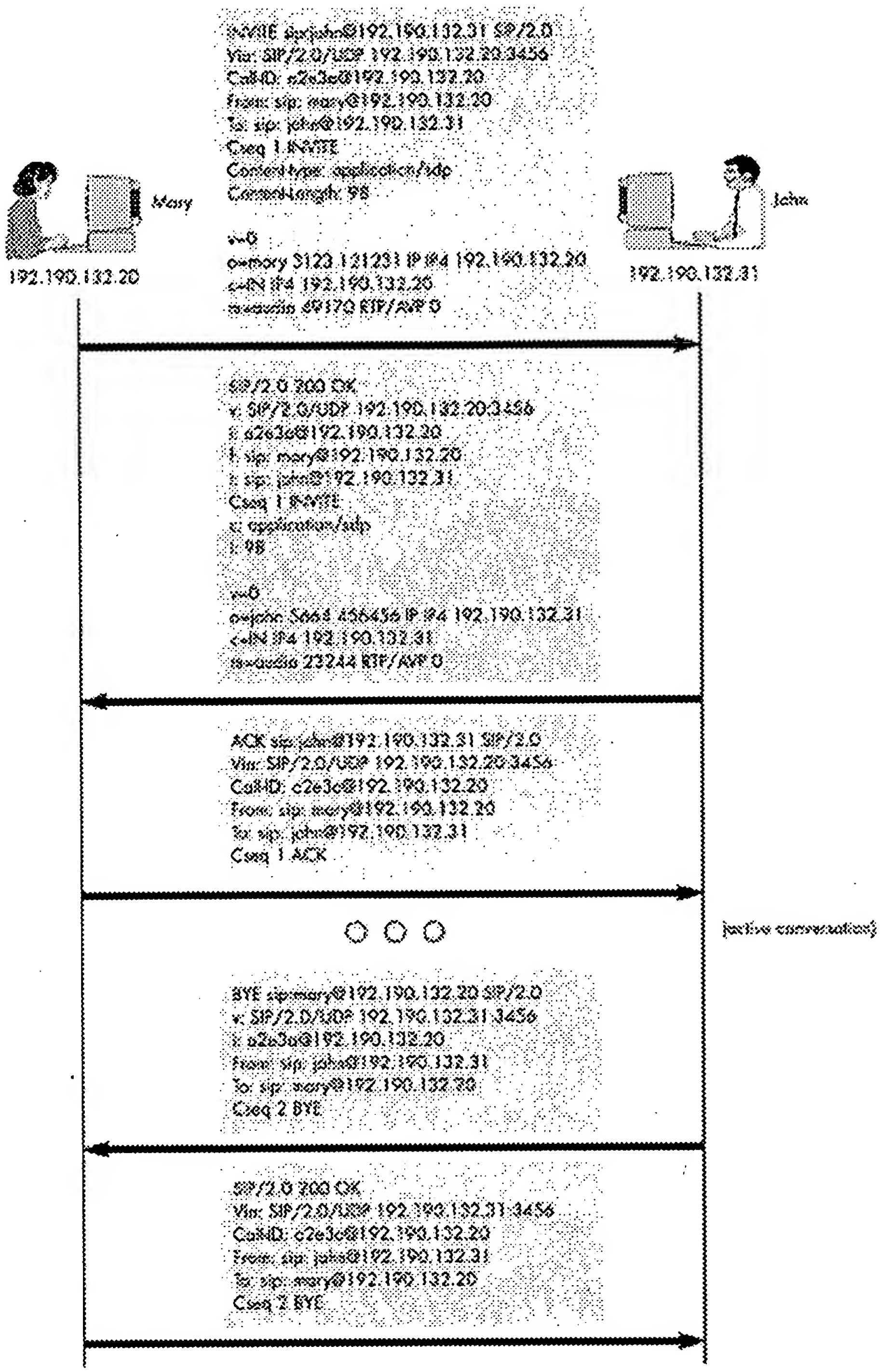
SIP does not have a notion of logical channels (as defined in H.323). When a client offers to receive several types of media on several UDP or TCP ports, it must be prepared to immediately receive media on any of those ports. However, the called terminal may choose to send data on some ports only (e.g. it does not have video capabilities and sends audio

only). Nothing in the signaling tells the client whether a port is going to be active or not. In general, this is not really a problem as most endpoints can keep listening to unused ports without significant performance impact. In some cases, however, SIP entities need to maintain many media channels and have to reuse UDP ports as efficiently as possible – this is the case for large gateways or centralized media resources. Those entities may have to multiplex several media channels on a single port, or close idle ports based on heuristics, for instance after a time-out period with no activity.

Terminating a call

The above example is a simple and successful call setup. Here we look at the complete call signaling, including the call termination by John (Fig. 2.3). If Mary had terminated the call, she would have sent the BYE request, and the From and To fields would be reversed. The media flows are not shown, but the signaling messages include all mandatory headers.

Figure 2.3. Complete call signaling



Some SIP headers have abbreviated forms that can help in keeping the total size of a message below the MTU. In this example John's terminal is using the abbreviated form.

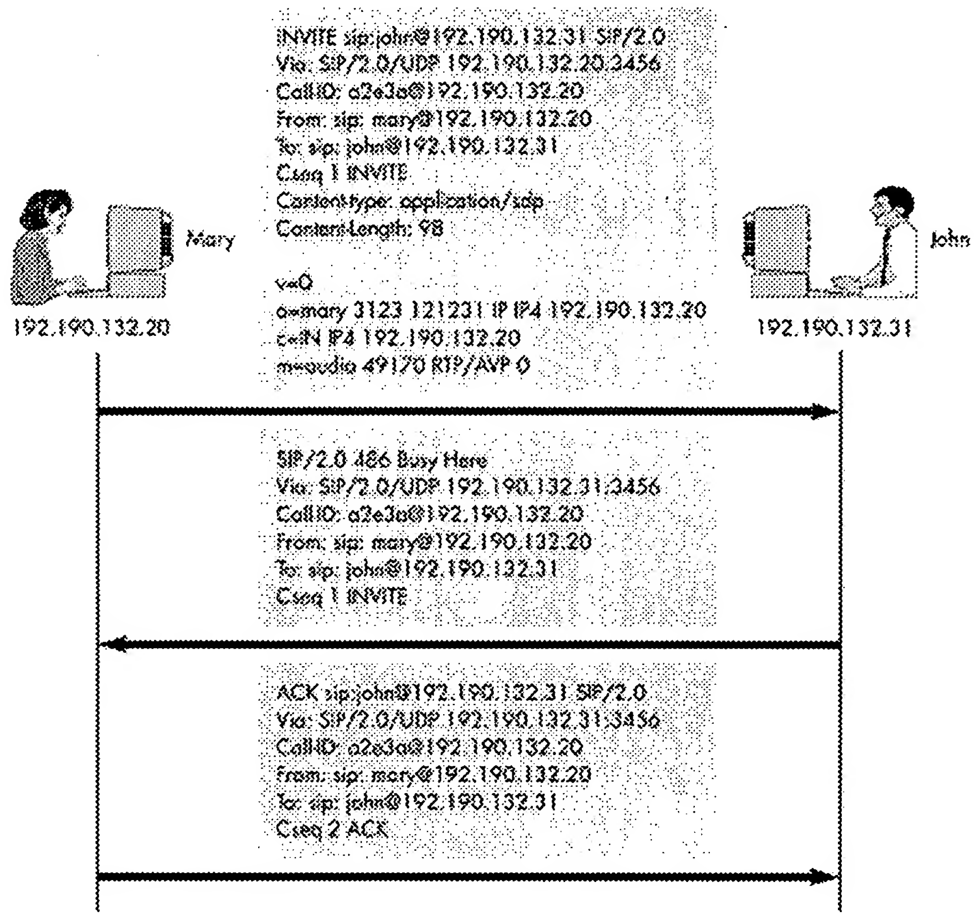
Rejecting a call

John can be unable to receive a call from Mary for a number of reasons. He may not be there or may be unwilling to answer, or he may be in another conversation. These conditions can be expressed in the reply message. SIP provides codes for the usual conditions, but also defines more sophisticated replies, such as 'gone', 'payment required' or

'forbidden'.

A simple 'busy here' reply (Fig. 2.4) tells Mary that John cannot be reached at this location (but she might try to reach another location, such as John's mobile phone through a gateway, or a voice mail). Another reply, '600 busy everywhere', will tell her that John cannot be reached at any location at this moment.

Figure 2.4. The 'busy here' reply

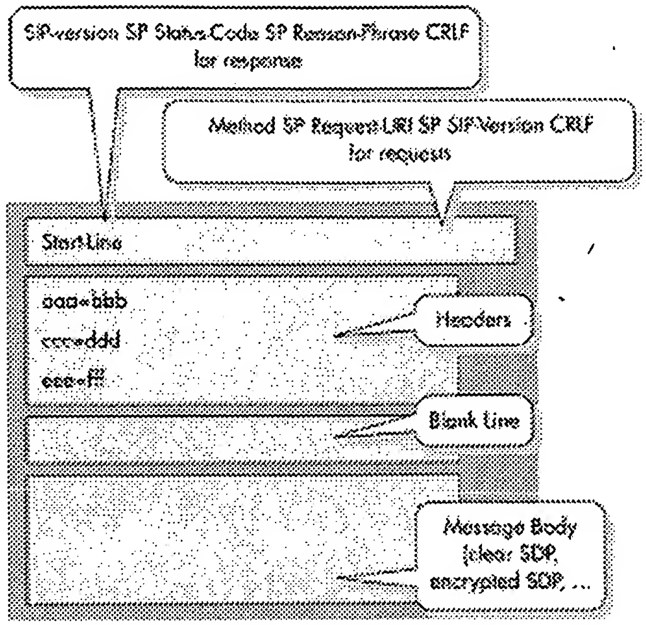


SIP messages

SIP messages are encoded using HTTP/1.1 message syntax (RFC 2068). The character set is ISO 10646 with UTF-8 encoding (RFC 2279). Lines are terminated with CRLF (carriage return, line feed), but receivers should be able to handle CR or LF as well.

There are two types of SIP messages: Requests and Responses. They share a common format, shown in Fig. 2.5 (where SP is an abbreviation for 'single space').

Figure 2.5. The SIP message format



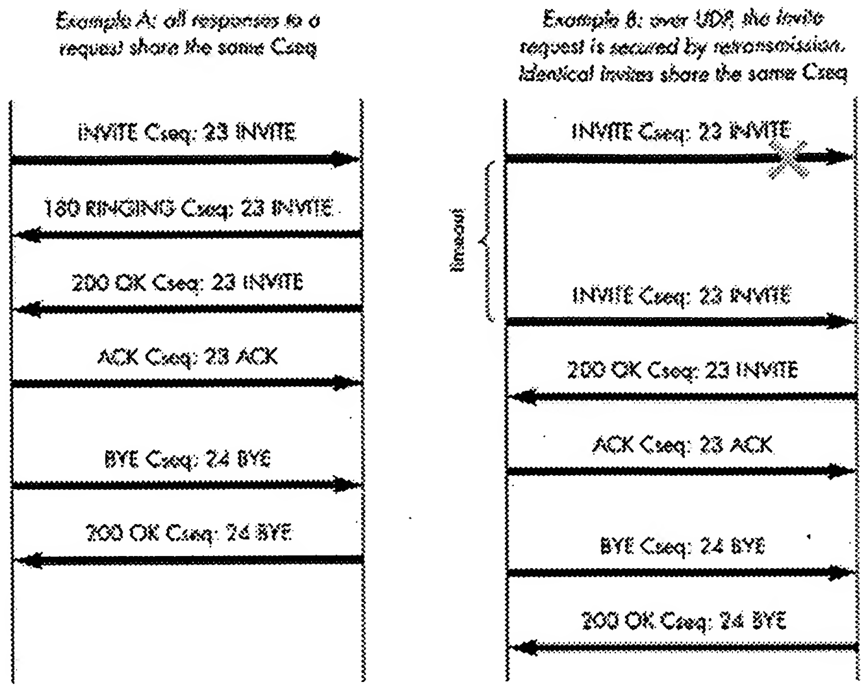
Some of the header fields are present in both requests and answers. They are part of the 'general header':

- *Call-ID* (e.g. 'CallID: f81d4fae-7dec-11d0-a765-00a0c91e6bf6@foo.bar.com'). The CallID parameter serves many purposes. In Register and Options requests it serves to match requests with the corresponding responses. For the Invite and Registration requests it also helps to detect duplicates (duplicate invite requests can occur when there is a forking proxy in the path). Successive Invite requests with the same CallID but different parameters can be used to dynamically change parameters in a conference.

  The first part of the CallID is meant to be unique within each host, and the last part, a domain name or host IP address, makes it globally unique (in case an IP address is used, it must be routable, i.e. private addresses such as 10.x.x.x cannot be used). A new CallID must be generated for each call;

- *Cseq* (e.g. Cseq: 1234 Invite). Every request has to have a Cseq header field, which is composed of an unsigned sequence number and the method name. Within a call, the sequence number is incremented at each new request (unless the request is a retransmission of a strictly identical previous request), and starts at a random value. The only exceptions are the ACK and Cancel requests which keep the Cseq number of the acknowledged reply (for ACK) or the cancelled request (for Cancel). The server must copy the Cseq value of the request in the corresponding replies. *See* Fig. 2.6.

Figure 2.6. Cseq usage

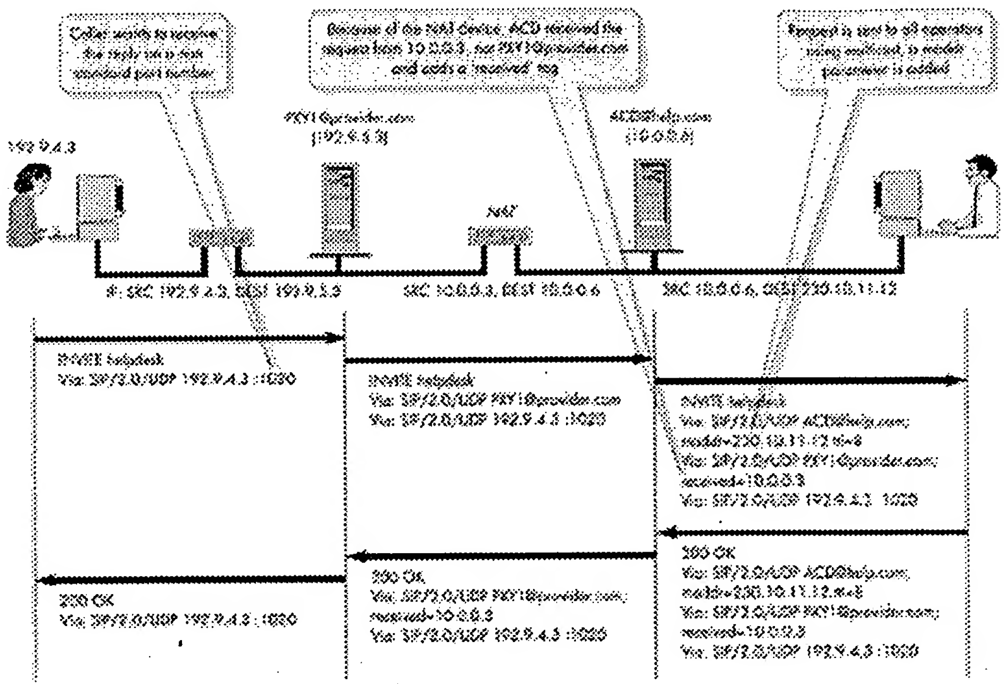


- *From* (e.g. From: 'MyDisplayName'<sip:myaccount@company.com>). This field must be present in all requests and responses. It contains an optional display name and the address of the originator of the request. Optional tags can be appended. Note that the 'from' field contained in SIP replies is simply copied from the request and therefore does not designate the originator of the reply;

- *To* (e.g. To: Helpdesk <sip:helpdesk@company.com>;tag=287447). This field must be present in all requests and responses and indicates the intended destination of a request. It is simply copied in responses. The tag is used mainly when a single SIP Uniform Resource Identifier (URI) designates several possible endpoints (as in the case of a helpdesk.) In this case a random tag is appended in the replies to allow a client to distinguish replies from individual endpoints;

- *Via* (e.g. Via: SIP/2.0/UDP PXY1.provider.com>; received 10.0.0.3). The Via field is used to record the route of a request, in order to allow intermediary SIP servers to forward the replies along the same path. *See* Fig. 2.7.

Figure 2.7. The Via field



In order to achieve this, each proxy adds a new Via field with its own address to the list of existing Via fields. The request receiver can add optional parameters, for instance to indicate that it received the request from an address that is not the address contained in the previous hop's Via field. Using this information, a proxy can forward the replies to the original sender, even if there is a network address translation device in the path. In the above example, the call center uses only private addresses (10.x.x.x), and is connected to the Internet through a router doing network address translation. When it receives the Invite request from proxy PXY1, the request's source IP address is not 192.9.5.3, but has been changed by the NAT device to 10.0.0.3. So the Automatic Call Distribution (ACD) system records this information in a 'received' parameter. On receiving the reply for this query, it will know that it must forward the request to 10.0.0.3, not the address of PXY1@provider.com. This mechanism also works when sending a request out of an IP domain using non-routable addresses.

This example also shows how multicast has been taken into account. When a maddr parameter is present in a Via field, the reply is forwarded in multicast using the maddr address (and the ttl value stored in the ttl parameter).

The Via header field is one of the most powerful features of SIP, and shows that SIP has been designed with IP networking issues in mind;

- *Encryption* (e.g. Encryption: PGP version=2.6.2,encoding=ascii). This header field specifies that the message body, and possibly some message headers, have been encrypted. For more detail see the security section further on.

Some header fields apply directly to the message body and are part of the 'entity header':

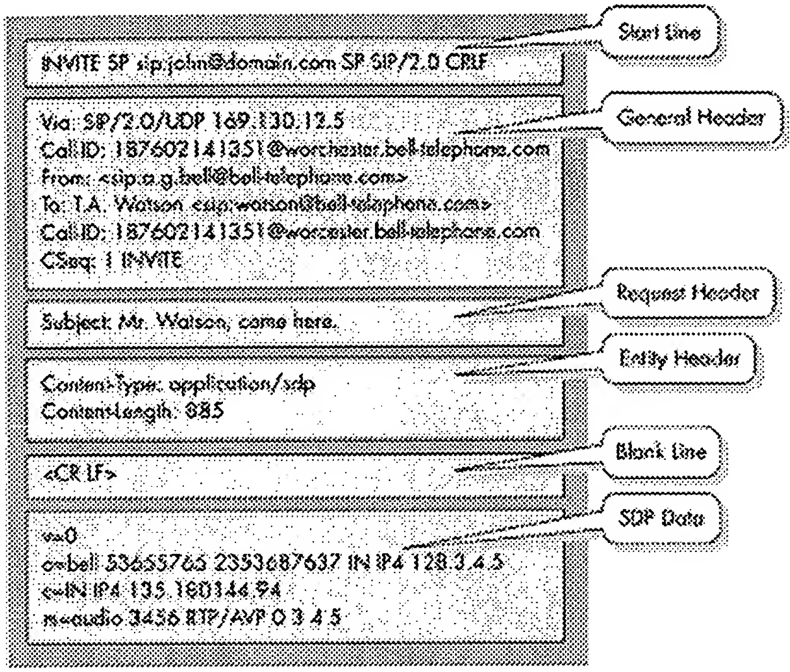
- *Content-Type* (e.g. Content-Type: application/sdp). This describes the media type of the content of the message body. In the example, the message body contains a session description using the IETF SDP protocol. Another example is text/html;

- *Content-length*. The number of octets of the message body.

SIP requests

SIP requests (Fig. 2.8) are sent from the client terminal to the server terminal.

Figure 2.8. The SIP Request message format



There are various methods for doing this:

- *ACK*. An ACK request is sent by a client to confirm that it has received a final response from a server, such as 200 OK;
- *BYE*. A BYE request is sent either by the calling agent or by the caller agent to abort a call;
- *Cancel*. A Cancel request can be sent to abort a request that was sent previously as long as the server has not yet sent a final response;
- *Invite*. The Invite request is used to initiate a call;
- *Options*. A client sends an Option request to a server to learn its capabilities. The server will send back a list of the methods it supports. In some cases it may also reply with the capability set of the user mentioned in the URL (uniform resource locator), and how it would have responded to an invitation;
- *Register*. Clients can register their current location (one or more addresses) with the Register request. A SIP server that can accept a Register message is called a registrar.

In addition to the fields of the general header, requests can carry fields in the request header:

- *Accept* (e.g. Accept: application/sdp, text/html). This optional header indicates what media types are acceptable in the response. The syntax is specified in RFC 1288;
- *Accept-Language* (e.g. Accept-Language: fr, en-gb;q=0.8, en;q=0.7). This indicates the preferred languages of the caller. The syntax is specified in RFC 1288;

- *Expires* – Invite and Register – (e.g. Expires: Thu, 01 Dec 2000 16:00:00 GMT *or* Expires: 5 in seconds). For a Register message, this header field indicates for how long the registration will be valid. The registrar can shorten the desired value in its reply. For an Invite message, this can be used to limit the duration of searches;

- *Priority* (e.g. Priority: emergency). The values are those of RFC 2076, plus 'emergency';

- *Record-Route* – also a response header field – (e.g. Record-Route: sip:acd.support.com;maddr=192.190.123.234,sip:billing.netcentrex.net;maddr=192.194. 126.23). Some proxies may add/update this header field if they want to be on the path of all signaling messages. In the example the request has traversed a billing proxy, then an ACD. Such entities need to monitor the state of calls in order to work properly. For example, the billing server might control a firewall to enforce the billing policy;

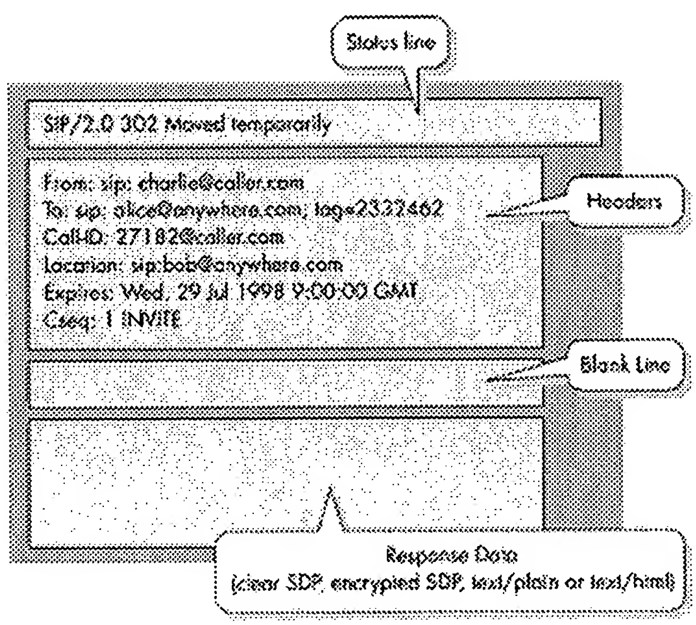
- *Subject* (e.g. Subject: 'Conference call on the SIP chapter'). This is free text that should give some information on the nature of the call.

SIP responses

A SIP server responds to a SIP request with one or more SIP responses. Most responses (2xx, 3xx, 4xx, 5xx and 6xx) are 'final responses' and terminate the SIP transaction. The 1xx responses are 'provisional' and do not terminate the SIP transaction.

The first line of a SIP response always contains a status code and a human readable reason phrase. Most of the header section is copied from the original request message. Depending on the status code, there may be additional header fields, and the response data part may be empty, or it may contain SDP data or explanatory text. *See* Fig. 2.9.

Figure 2.9. The SIP response format



So far six categories of status codes have been defined, depending on the first digit as shown in Table 2.1.

This classification makes it easier to add new status codes. When an old terminal does not understand a new CXX code, it should treat it as a C00 code. Therefore even old terminals will be able to react 'intelligently' when facing unknown status codes. These terminals can also give some additional information to the user if a reason phrase is present.

Table 2.1. The six categories of status codes

1xx	Informational		Request received, continuing to process the request
		100	Trying
		180	Ringing
		181	Call is being forwarded
		182	Queued
2xx	Success		The action was successfully received, understood, and accepted
		200	OK
3xx	Redirection		Further action must be taken in order to complete the request
		300	Multiple choices
		301	Moved permanently
		302	Moved temporarily
		380	Alternative service
4xx	Client error		The request contains bad syntax or cannot be fulfilled at this server
		400	Bad request
		401	Unauthorized
		402	Payment required
		403	Forbidden
		404	Not found
		405	Method not allowed
		406	Not acceptable
		407	Proxy authentication required
		408	Request timeout
		409	Conflict
		410	Gone
		411	Length required
		413	Request message body too large
		414	Request-URI too large
		415	Unsupported media type
		420	Bad extension
		480	Temporarily not available
		481	Call leg/transaction does not exist
		482	Loop detected
		483	Too many hops
		484	Address incomplete
		485	Ambiguous

5xx	Server error	The request contains bad syntax or cannot be fulfilled at this server	
		500	Internal server error
		501	Not implemented
		502	Bad gateway
		503	Service unavailable
		504	Gateway timeout
		505	SIP version not supported
6xx	Global failure		The request is invalid at any server
		600	Busy everywhere
		603	Decline
		604	Does not exist anywhere
		606	Not acceptable

Session description syntax, SDP

SIP uses the session description protocol (SDP) specified in RFC 2237. SDP is also a product of the MMUSIC working group, and is heavily used today in the context of the MBONE, the multicast enabled overlay network of the Internet.

In order to be able to receive an MBONE session, a receiver needs to know:

- which multicast address is going to be used by the session
- what will be the UDP destination port
- the audio and/or video coders that will be used (GSM, H.261, etc.)
- some information on the session (name, short description)
- contact information
- activity schedule.

The primary purpose of SDP is to define a standard syntax for this type of information.

The SDP session description can be conveyed using a variety of transport methods, depending on the context:

- the session announcement protocol (SAP) on the MBONE
- the real-time streaming protocol (RTSP) for streaming applications
- SIP to set up point-to-point and multipoint communications.

SDP is a human readable protocol, consisting of several <type>=<value> lines terminated by CRLF. The field names and attributes use US-ASCII characters, but free text fields can be localized since SDP uses the complete ISO 10646

character set. This philosophy, as opposed to a binary encoding like ASN-1 PER, facilitates programming and debugging at the expense of a greater bandwidth usage. However, the bandwidth usage of a signaling protocol is negligible compared to the actual media flows, so the trade-off is very good.

The session description is structured in one section which applies to the whole session (starting with v= ...), and several media description sections (each starting with (m= ...). Parameters in the media sections can override the default parameters of the session level section.

Table 2.2 describes the various field types defined by RFC 2237 for each section.

Table 2.2. Field types described by RFC 2237

Session level field type	Sub-section level field type	Usage	Format and example	
v=		protocol version	v=0	M
o=		owner/creator and session identifier	o=<username> <session id> <version> <network type> <address type> <address> o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4	M
s=		session name	s=<session name> s=SDP Seminar	M
i=		session information	i=<free text session description> i=A Seminar on the session description protocol	O
u=		URI of description	u=<Universal Resource Identifier> u=http://www.cs.ucl.ac.uk/staff/M.Handley/ sdp.03.ps	O
e=		email address	e=<email address> (Optional free Text) or e=<Optional free Text> "<"email address">" e=mjh@isi.edu (Mark Handley) e= Mark Handley <mjh@isi.edu>	O
p=		phone number	p=<phone number> (Optional free Text) or p=<Optional free Text> "<"phone number">" p=+44-171-380-7777	O
c=		connection information – not required if included in all media	c=<network type> <address type> <connection address> TTL must be included for multicast sessions. c=IN IP4 224.2.17.12/127	O

			c=IN IP4 224.2.1.1/127	
b=		bandwidth information	b=<modifier (CT Conference Total|AS Application-Specific Maximum>:<bandwidth-value in kilobits/s> b=CT:120	O
One or more time description sections	t=	time the session is active	t=<start time> <stop time>, using decimal NTP in seconds t=2873397496 2873404696	M
	r=	zero or more repeat times	r=<repeat interval> <active duration> <list of offsets from start-time>, by default in seconds r=604800 3600 0 90000 means that the repeat interval is 1 week (604800 seconds), active for one hour (3600 seconds) after each offset from the start time T. Offsets are here 0 seconds and 90000 seconds (25 hours). I.E, if *** represents active periods and --- idle periods: T***T+1h---T+25h***T+26h-----T+1week****T+1week+1h-----T+1Week+25h*** ... The repetition is valid until the stop time. Unit modifiers can be used for compactness, and the previous record can also be written as follows: r=7d 1h 0 25h	O
z=		time zone adjustments		O
k=		encryption key	k=<method>:<encryption key> or k=<method>	O
a=		zero or more session attribute lines	a=<attribute> or a=<attribute>:<value> a=recvonly	O
Zero or more media descriptions	m=	media name and transport address	m=<media> <port> <transport> <format list> m=audio 49170 RTP/AVP 0 3 means that the media is audio, can be received on port 49170 (RTP only uses even ports, the next odd port ring being used by RTCP). The transport is protocol RTP/AVT (IETF's realtime transport protocol using the audio/video profile carried over UDP), and the format is media payload types 0 or 1 of the AVT profile (0 is μ-law PCM coded single channel audio sampled at 8KHz, 3 is GSM) Other RTP profiles would be coded after the slash, e.g. a hypothetical profile XXX would appear as RTP/XXX.	M
	i=	media title		O
	c=	connection information –		

	optional if included at session level	
b=	bandwidth information	O
k=	encryption key	O
a=	zero or more media attribute lines	O

Dynamic and static payload types

Under a particular profile, some RTP payload types are static: i.e. their meaning is fully defined in the profile (e.g. RTP/AVP 0 is 64 kbit/s µ-LAW PCM). Other RTP payload types have a meaning only in association with a particular session described in SDP. These are dynamic payload types. The SDP RFC gives the example of the 16-bit linear encoded stereo audio sampled at 16 kHz. There is no payload type defined that would exactly correspond to this. Instead we will be using an arbitrary unused number for the payload type, say 98 (m=video 49232 RTP/AVP 98), and will describe the format of the transported data in SDP:

```
a=rtpmap:98 L16/16000/2
```

The format is a=rtpmap:<payload type> <encoding name>/<clock rate>[/<encoding parameters>], which in our case translates to 16 linear, 16 000 Hz sampling, two channels.

By extension, the term 'dynamic payload type' applies to any RTP stream for which the media encoding characteristics are conveyed out of band (for instance through an H.245 OpenLogicalChannel message).

URL http://proquest.safaribooksonline.com/0201619105/ch02lev1sec1

Additional reading

Safari has identified sections in other books that relate directly to this selection using Self-Organizing Maps (SOM), a type of neural network algorithm. SOM enables us to deliver related sections with higher quality results than traditional query-based approaches allow.

Section Title	Book Title
1. Sample SIP Call Message Flow	Practical VoIP Using VOCAL By Luan Dang, Cullen Jennings, David G. Kelly
2. What Is SIP?	Practical VoIP Using VOCAL By Luan Dang, Cullen Jennings, David G. Kelly
3. Session Initiation Protocol (SIP)	Network Consultants Handbook By Matthew J. Castelli
4. Session Initiation Protocol (SIP)	Internetworking Technologies Handbook, Fourth Edition By Cisco Systems, Inc.
5. Message Headers	Practical VoIP Using VOCAL By Luan Dang, Cullen Jennings, David G. Kelly
6. VoIP Signaling: SIP Paradigm	Integrating Voice and Data Networks By Scott Keagy
7. SIP	Switching to VoIP

8. General SIP Tutorial
9. Inspect the SIP Message Structure
10. High-Level Design

By Theodore Wallingford

Internetworking Technologies Handbook, Fourth Edition
By Cisco Systems, Inc.

VoIP Hacks
By Theodore Wallingford

Practical VoIP Using VOCAL
By Luan Dang, Cullen Jennings, David G. Kelly